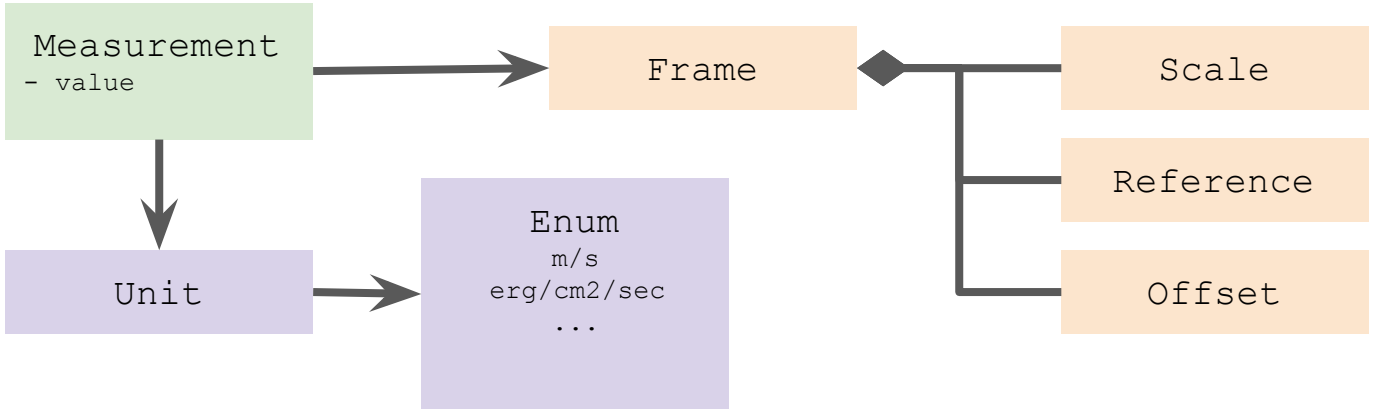
Data Models in the VO

Status - Purpose - Prospects

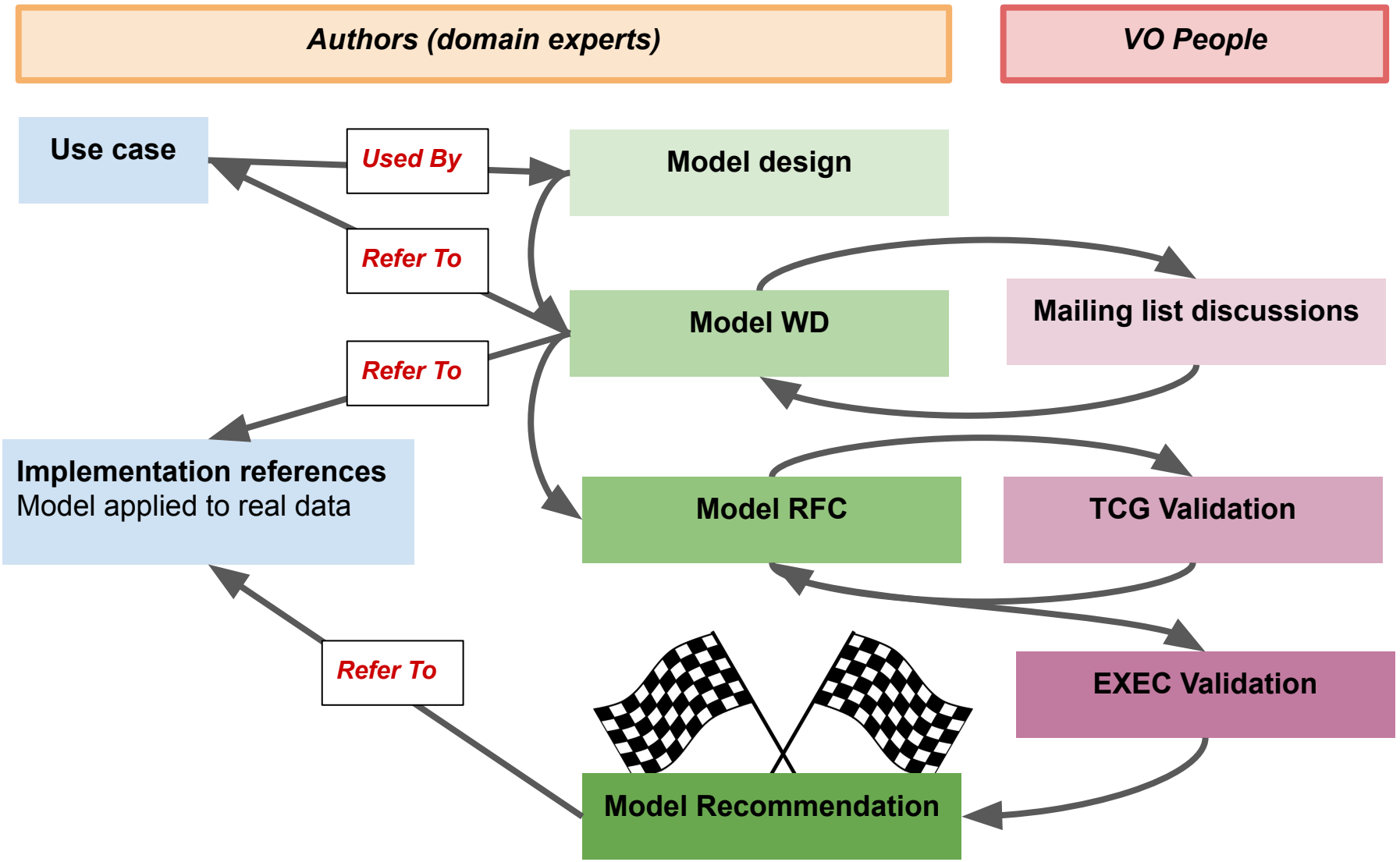
Laurent Michel, Jesus Salgado -- Locktown - 2020

What is a Data Model

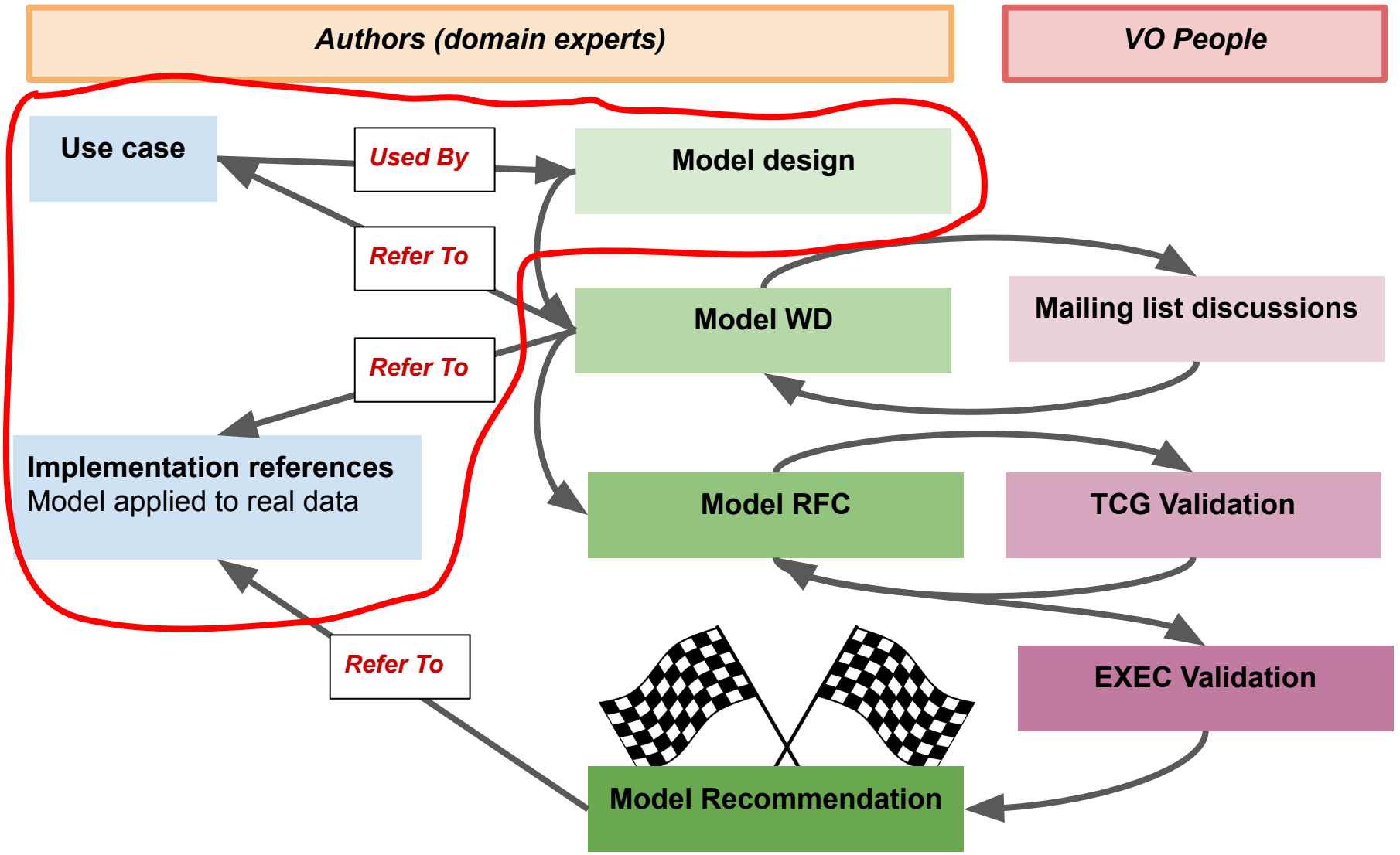
- **Formal description of the quantities used by the experts in a domain**
- **What does the human knowledge (common sense) say:**
  - A **`measurement`** is a value with a given **`unit`** that is valid in a given **`frame`**
- **The Model gives a formal representation of that knowledge**
  - The model defines the quantity classes, the names, the vocabulary and ….
    …. The relationships between those elements



Design Process for VO Models



Design Process for VO Models



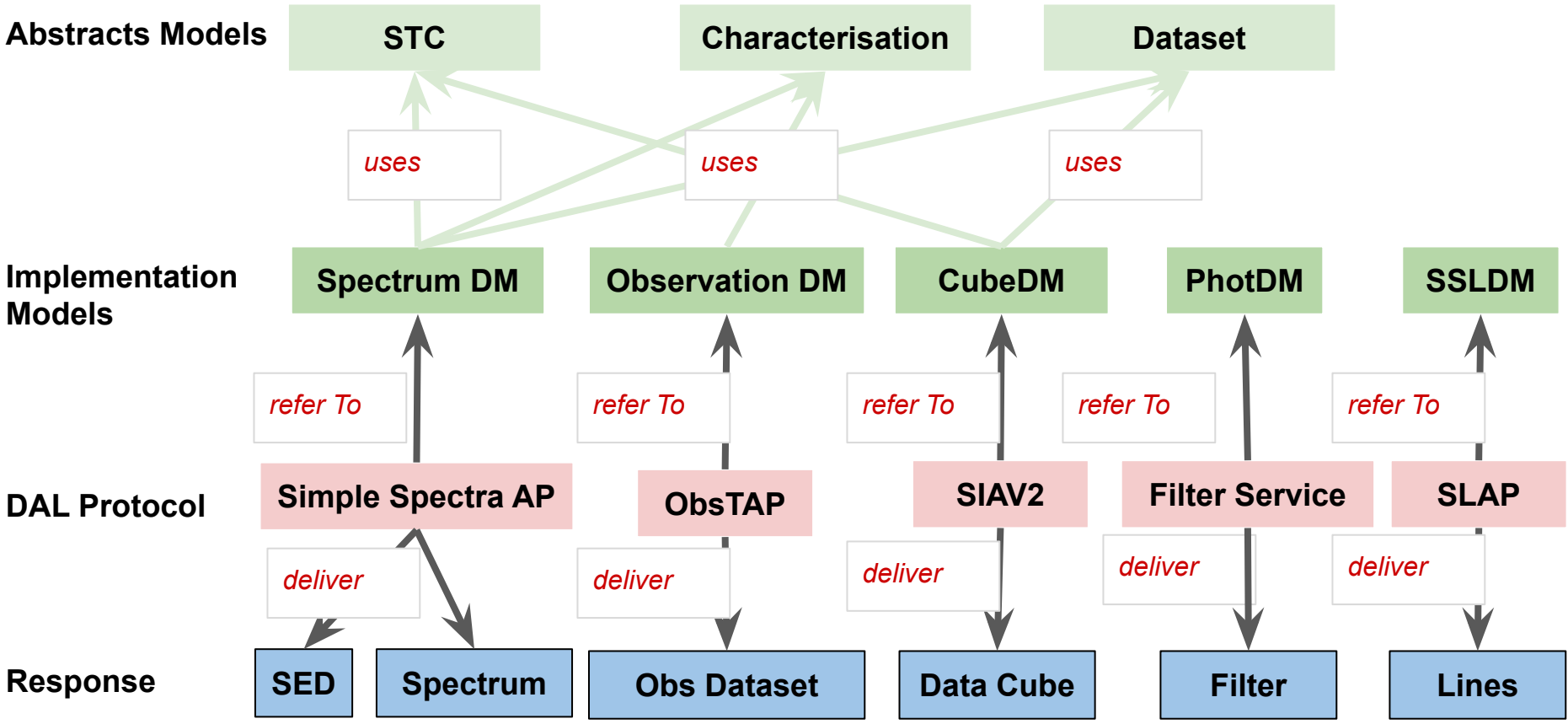
Models Recommended by the VO

STC	1.33	2007	Explain and document the design and implementation of the Space-Time (and related coordinate axes) metadata for the Virtual Observatory.
Characterization	1.13	2008	Characterisation is intended to define and organize all the metadata necessary to describe how a dataset occupies multidimensional space
SSLDM	1	2010	a simple framework, both for atomic and molecular line databases, as well as for databases of observed lines in all energy ranges, or for VO-tools,
Spectrum	1.1	2011	Representation of single 1-dimensional spectrum
Obscore	1	2011	Obscore addresses the problem of an astronomer posing a world-wide query for scientific data discovery
SimDM	1	2012	The goal of this model to support discovery of simulations by describing those aspects of them that scientists might wish to query on, i.e. it is a model for meta-data describing simulations
PhotDM	1	2013	This document outlines a photometry data model to describe photometric measurements in a standard way.
Obscore	1.1	2017	See V1 above
VO-DML	1.0	2018	Standard modelling language, or meta-model, for expressing data models in the IVOA.
Provenance	1.0	2020	Model for provenance information used to enable any scientist to trace back the origin of a dataset
SSLDM	2.0	2020?	See V1 above

What Are VO Data Models Used For

- **Documentation**
  - Developer guideline
    - Developers work with the DM standard on the table
    - Client, server, validator
  - DAL protocol design
    - Designing protocol where data responses are retrospectively compliant with a model

Documentation: Binding DAL protocols with models



Responses are all VOTables
VOTable fields and params are defined by the DAL protocol
They match the model by construction

What Are VO Data Models Used For

- **Documentation**
  - Developer guideline
    - Developers work with the DM standard on the table
    - Client, server, validator
  - DAL protocol design
    - Designing protocol where data responses are retrospectively compliant with a model

- **Interoperability**
  - **Different data mapped on the same model can be combined or compared to each other**
    - **Data discovery (Obscore)**
    - **Stacked plots**
    - **Cross-match**

Interoperability: Data Models and VO-Tables

- **VOTables are containers**
  - A generic VOTable schema can validate the XML structure of the container
  - It cannot validate the content of the container
- **VOTable schema useless to process models**
  - Cannot say how data in a VOTable are mapped on a model
  - Cannot even say whether data in a VOTable match a given model
- **Not a problem for VOtables delivered by simple DAL protocols**
  - The VOTable structure is defined by the protocol
- **Big problem for VOTable containing native data**
  - Vizier, TAP
  - How to bind native data with a given model
  - This is a key point for interoperability

Validating VOTables with models

```
<MODEL> (informal)
  <POSITION>
      Something named ra [0 360]
      Something named dec [-90 +90]
  <POSITION>
</MODEL>
```

Validating VOTables with models

```
<MODEL> (informal)
  <POSITION>
      Something named ra [0 360]
      Something named dec [-90 +90]
  <POSITION>
</MODEL>
```

```
<VOTABLE>
  <TABLE>
    <FIELDS name="ra"/>
    <FIELDS name="dec"/>
    <DATA>
     <TR>
        <TD>12.34</TD>
        <TD>-78.65</TD>
      </TR>
    </DATA>
  </TABLE>
</VOTABLE>
```

Validating VOTables with models

```
<MODEL> (informal)
  <POSITION>
      Something named ra [0 360]
      Something named dec [-90 +90]
  <POSITION>
</MODEL>
```

```
<VOTABLE>
  <TABLE>
    <FIELD name="ra"/>
    <FIELD name="dec"/>
    <DATA>
     <TR>
        <TD>12.34</TD>
        <TD>-78.65</TD>
      </TR>
    </DATA>
  </TABLE>
</VOTABLE>
```

```
<VOTABLE>
  <TABLE>
    <FIELD name="RA2000"/>
    <FIELD name="DE2000"/>
    <DATA>
     <TR>
        <TD>12.34</TD>
        <TD>-78.65</TD>
      </TR>
    </DATA>
  </TABLE>
</VOTABLE>
```

Both VOTables are valid against the VOTable schema
One is not compliant with a (trivial) model for sky position
Not generic tool in the VO-DM landscape to achieve this semantic validation

Actual Solution : Annotating Data with UTypes

```
<MODEL> (informal)
  <POSITION>
      Ra is Something tagged with position.long
      Dec is Something tagged with position.lat
  <POSITION>
</MODEL>
```

```
<VOTABLE>
  <TABLE>
    <FIELDS name="ra" utype="position.long"/>
    <FIELDS name="dec" utype="position.lat/>
    <DATA>
     <TR>
        <TD>12.34</TD>
        <TD>-78.65</TD>
      </TR>
    </DATA>
  </TABLE>
</VOTABLE>
```

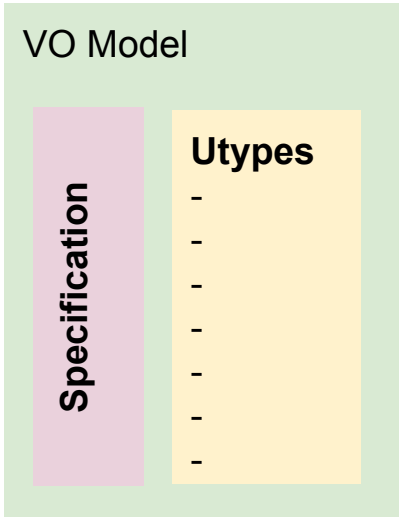
```
<VOTABLE>
  <TABLE>
    <FIELDS name="RA2000" utype="position.long"/>
    <FIELDS name="DE2000" utype="position.lat/>
    <DATA>
     <TR>
        <TD>12.34</TD>
        <TD>-78.65</TD>
      </TR>
    </DATA>
  </TABLE>
</VOTABLE>
```

Both VOTables are valid against the VOTable schema
Both are compliant with a (trivial) model for sky position
Semantic can be checked against model tags (UTypes)

Suitable approach,
Allow to check that all quantities needed to build a model instance are available

Utype Weakness

- **No common way to describe models**
  - Model are not machine readable
  - No standard set of modeling features supported by the VO



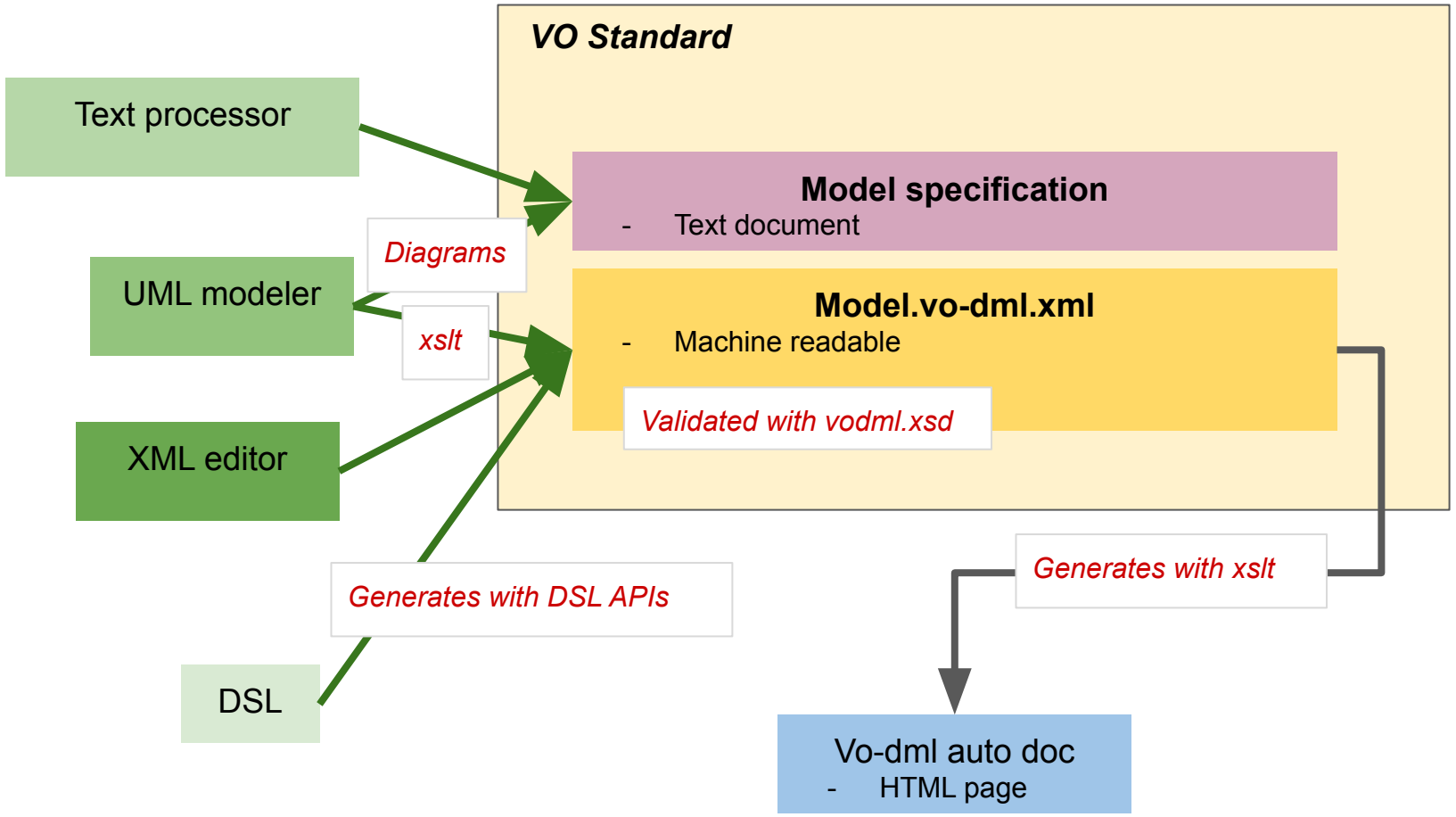
- **Data annotation based on Utypes not totally satisfactory**
  - No common rule for UTypes definition
  - No common rule for parsing UTypes
    - The way to parse UTypes is model-specific
  - Possible duplication of UTypes
    - E.g. Positions in the same model (source location raw and corrected)

VO-DML: a VO Meta-model (lead by G. Lemson)

- **VO-DML: a formal way to specify any VO models**
  - VO-DML is a meta-model
    - Restricted set of model features
      - No multiple inheritances
      - No object aggregation
      - No multiple compositions
  - Models serialized in XML files
    - Validation by a schema
  - *VO-DMLized* models are machine readable
    - Possibilty of applying XML style sheets to VO-DML files
      - Doc generation (part of the actual workflow)
      - Data template
      - Code generation
    - Possibility of formally import models
    - Possibility of developing common library and tools

- **VO-DML is a Recommendation since September 2018**
  - Any new model must be published as a VO-DML file
  - Former models must be serialized in VO-DML to be used in new models
    - This might require some changes in the model

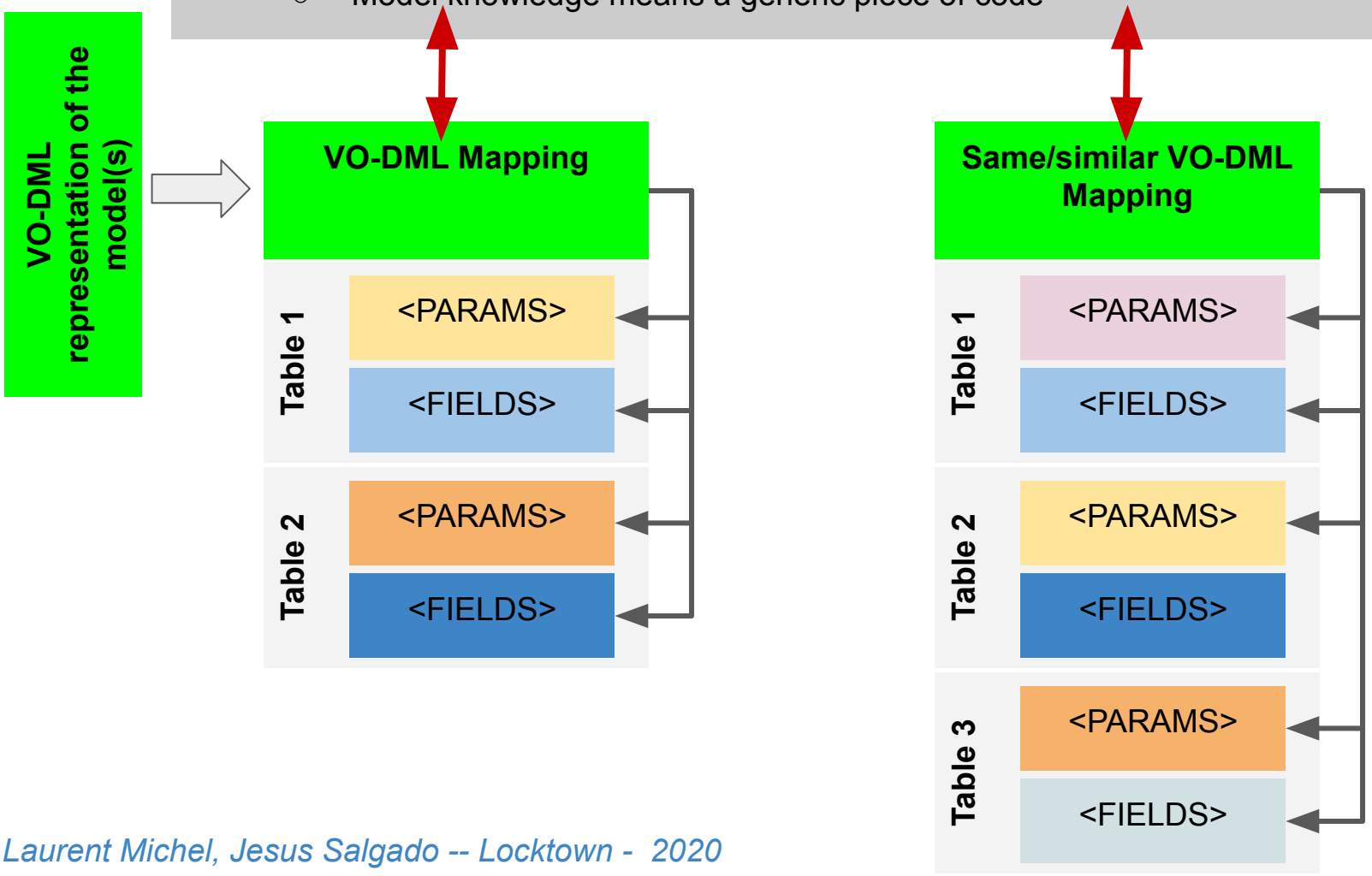
VO-DML : An Uniform Way to Formalize Models



Replacing UTypes with Data Annotation with VO-DML

VO-DML Aware Client

- Only sees the mapping block
- No need to search annotations within tables
- The model knowledge is sufficient to process any table instance
  - Model knowledge means a generic piece of code



Replacing UTypes with Data Annotation with VO-DML

Model

```
<VOTABLE>
  <VODML>
    <MODELS>...</MODELS>
    <GLOBALS>...</GLOBALS>
    <TEMPLATES tableref=t_1>
    <TEMPLATES tableref=t_2>
    ....
  </VODML>
  <RESOURCE>
      <TABLE ID=t_1>...</TABLE>
      <TABLE ID=t_2>...</TABLE>
  </RESOURCE>
</VOTABLE>
```

- A mapping block inserted on the top of the VOTable
- The mapping block is pure XML with a specific schema
- The mapping block reproduces the hierarchy of the model elements
- Model elements contain references to the table columns or to literal values
- This makes possible to build model instances from VOTable data
- Model mapping can be ignored

Replacing UTypes with Data Annotation with VO-DML

- **There is a working draft**
  - Tested on a various models
  - Rather chatty and complicated (personal opinion)
  - Hackathon in Victoria (May 2018)
    - https://wiki.ivoa.net/twiki/bin/view/IVOA/InterOpMay2018VODML
  - Tools
    - Graphical mapper (G. Lemson)
    - Model specific API in Python (O. Laurino)
    - MAST prototype (T. Donaldson)
  - *No big momentum in the community to use it yet*

- **There is another one with a simplified syntax**
  - By myself
    - https://github.com/lmichel/vodml-lite-mapping
  - Tested on both TimeSeries and CAB_MSD model drafts
  - Model-agnostic API
  - *No big momentum in the community to contribute*

Using Models for Interoperability

- **Data annotation with UTypes**
  - Data elements refer to model leaves
  - Data response kinked to models by a key mechanism

works well

Using Models for Interoperability

- **Data annotation with UTypes**
  - Data elements refer to model leaves
  - Data response kinked to models by a key mechanism

works well

- **Data annotation with VO-DML mapping**
  - Data response comes with a whole description of the model they refer to.
  - The client has enough material to build model instance from the data.

a bit stuck for now

Using Models for Interoperability

- **Data annotation with UTypes**
  - Data elements refer to model leaves
  - Data response kinked to models by a key mechanisim

works well

- **Data annotation with VO-DML mapping**
  - Data response comes with a whole description of the model they refer to.
  - The client has enough material to build model instance from the data.

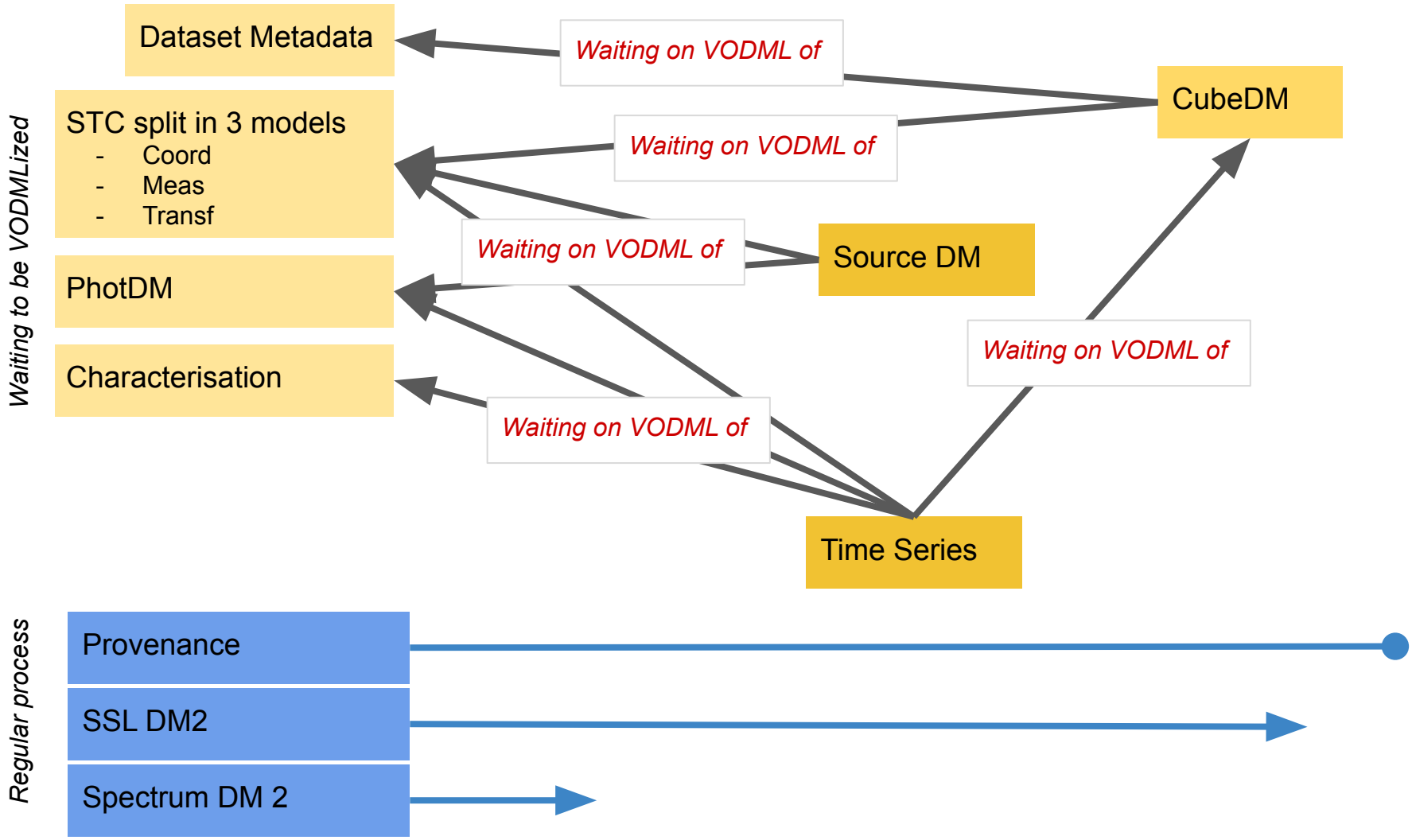
a bit stuck for now

- **Embedding models in client code**
  - Client code is enable to interpret (properly display e.g.) data just by analysing a model

rather a dream

Laurent Michel, Jesus Salgado -- Locktown - 2020

Ongoing Work



Thank you for your Attention

Laurent Michel, Jesus Salgado -- Locktown - 2020